CoRAL –
The Constrained RESTful Application Language

Klaus Hartke

CoRE is a vision for the *Internet of Things* where *constrained nodes* (or ‘things’) communicate with each other and the greater Internet in *REST style*.

As more and more things get deployed over time, it can be expected that the *Internet of Things* will undergo many **generations of – uncoordinated – changes** to which existing and new applications will need to adapt.

Hypermedia-driven **REST** provides a strategy to cope with change: the need to evolve as new use cases and technologies arrive.

It enables evolvability in two dimensions:

- the information model
  - evolvable media types describing the exchanged information
- the interaction model
  - adjustable hypermedia controls describing the interactions

See also: M. Kovatsch, Y. Hassan, K. Hartke. Semantic Interoperability Requires Self-describing Interaction Models. IoT Semantic Interoperability Workshop, 2016.

However, there are costs associated with the use of *hypermedia-driven* **REST**:

- higher design effort
  - designers do not only have to take current requirements into consideration, but also have to anticipate changes that may be required in the future
- more roundtrips
  - the incremental discovery of resources in hypermedia-driven applications may lead to a higher number of roundtrips
- larger representation size
  - the hypermedia controls included in representations can be very verbose and unnecessarily repetitive

Can we lower the costs of *hypermedia-driven* **REST** so that the benefits outweigh the costs?

Let’s take a look at representation size!

Link Format

```
</sensors/temp>;if=core.s,
</sensors/light>;if=core.s
```

- Provides a representation format for links
- All links have the same semantics:
  "There exists a resource with these attributes."
- The '`if`' attribute indicates both the information model and the interaction model of the resource, which includes the content format, supported methods, parameters, and more (see *I-D.ietf-core-interfaces-04*)
- Text-based format based on RFC 5988

» Quite compact, but not very expressive

HTML5

Links:

```
<a href="about.html">...</a>
<link rel="stylesheet" href="style.css">
```

Templated Links:

```
<form method="get" action="search.php">
  <input id="query" type="text">
</form>
```

Embedding Links:

```
<img src="logo.png">
<audio src="audio.ogg">
<video src="video.mp4">
```

Forms:

```
<form method="post" action="">
  <input id="name" type="text">
  <input id="age" type="text">
  <input id="homepage" type="text">
</form>
```

Hypermedia Formats

HTML5

- Provides a representation format for text, links (`<a>`, `<link>`), templated links (`<form method="get">`), embedding links (`<img>`, `<audio>`, `<video>`) & forms (`<form method="post">`)
- Generally depends on a human user to infer the semantics of links and forms (with the exception of `rel` attributes like `"stylesheet"`, `"prefetch"` and `"nofollow"`)
- Text-based format based on SGML

» Very expressive, but also very verbose and not machine-understandable

Hypertext Application Language (HAL)

```
{
    "_links": {
        "self": { "href": "/orders" },
        "next": { "href": "/orders?page=2" },
        "http://example.com/rels/find": {
            "href": "/orders{?id}",
            "templated": true
        }
    },
    "currentlyProcessing": 14,
    "shippedToday": 20,
    "_embedded": {
        "http://example.com/rels/order": {
            "_links": {
                "self": { "href": "/orders/123" },
            },
            "total": 30.00,
            "currency": "USD",
            "status": "shipped"
        }
    }
}
```

Hypertext Application Language (HAL)

- Provides a representation format for data, links (`"_links"`), templated links & embedded representations (`"_embedded"`) but no forms
- Machine-understandable semantics provided by *Link Relation Types (RFC 5988)*
- Text-based format based on JSON

» Very verbose, quite expressive

Thoughts

- I think we need both
  - very compact notations to express links and forms for common interaction patterns *(like Link Format)*, and
  - more explicit/verbose notations to express links and forms for less common interaction patterns and when things change *(like HTML & HAL)*
- Embedded representations seem to be really useful (see *I-D.hartke-core-lighting-00*)
- To reduce the representation size further: serialize links and forms in a compact, binary format; use numeric identifiers for media types, link relation types and form relation types

Constrained RESTful Application Language (CoRAL)

- Provides a representation format for Web links, forms and embedded representations
- Machine-understandable semantics provided by *Link Relation Types (RFC 5988)* and *Form Relation Types*
- Binary format based on CBOR

» Quite compact, very expressive

Links

The CoRAL representation of a resource contains a set of links where the context of each link is the represented resource.

For example, the Web link (in RFC 5988 syntax)

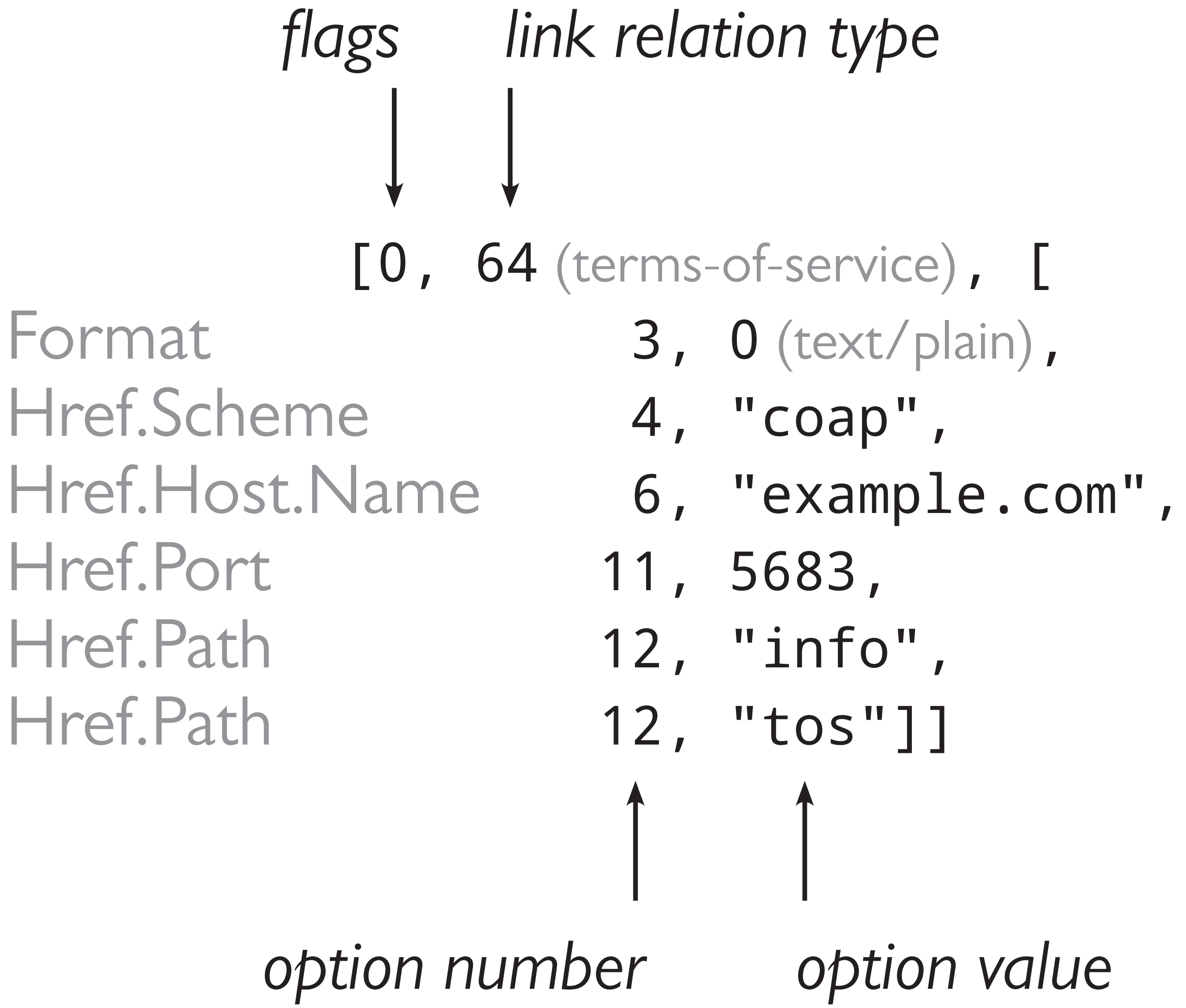
```
<coap://example.com:5683/info/tos>;
  rel=terms-of-service;
  type=text/plain
```

is serialized in CoRAL as follows:

```
[0, 64, [
           3, 0,
           4, "coap",
           6, "example.com",
          11, 5683,
          12, "info",
          12, "tos"]]
```

Data Format

Links are serialized as CoAP-style options, encoded in CBOR:



```
                      [0, 64 (terms-of-service), [
Format                          3, 0 (text/plain),
Href.Scheme                     4, "coap",
Href.Host.Name                  6, "example.com",
Href.Port                      11, 5683,
Href.Path                      12, "info",
Href.Path                      12, "tos"]]
```

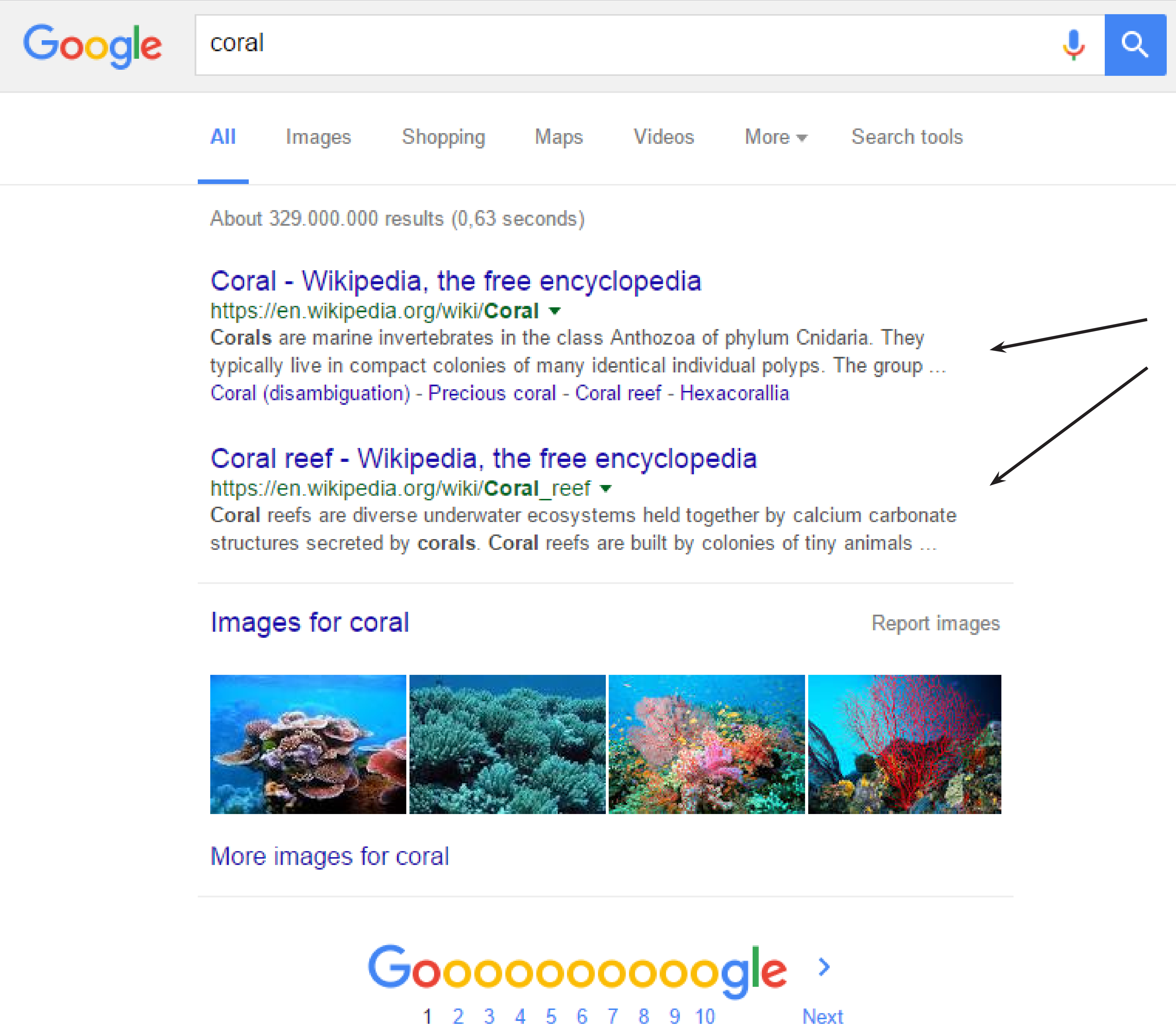
This alleviates the need to implement a full RFC 3986-compliant URI parser and resolver.

Embedding Representations

Links can embed a representation of the link target:

```
                    [4, 30 (item), [
Format                      3, 50 (application/json),
Href.Path                  12, "item1"],
Payload               h'7b20227461736b223a2022526574
                        75726e2074686520626f6f6b7320
                        746f20746865206c696272617279
                        222c202261737369676e6565223a
                        2022416c69636522207d']
```

```
{
   "task":     "Return the books to the library",
   "assignee": "Alice"
}
```



coral

All Images Shopping Maps Videos More Search tools

About 329.000.000 results (0,63 seconds)

Coral - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Coral**
Corals are marine invertebrates in the class Anthozoa of phylum Cnidaria. They typically live in compact colonies of many identical individual polyps. The group ...
Coral (disambiguation) - Precious coral - Coral reef - Hexacorallia

Coral reef - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Coral**_reef
Coral reefs are diverse underwater ecosystems held together by calcium carbonate structures secreted by **corals**. **Coral** reefs are built by colonies of tiny animals ...

Images for coral Report images

More images for coral

Goooooooooogle
1 2 3 4 5 6 7 8 9 10 Next

embedded representation of link target

Namespaces

The link relation type in a serialized link may be from the “global” or the “local” namespace.

The **global** namespace consists of the IANA-registered link relation types:

ID	Link Relation Type
1	about
2	alternate
3	appendix
4	archives
…	…

Namespaces

The **local** namespaces consists of link relation types defined by the +coral media type.

For example, a media type “application/example.shop+coral” could define the following set of local link relation types:

ID	Link Relation Type
-80	http://example.com/rels/order
-81	http://example.com/rels/basket
-82	http://example.com/rels/customer

Namespaces

Simiarly, a media type “application/example.foaf+coral” could define the following mapping from local link relation type IDs to the FOAF RDF model:

ID	Link Relation Type
-100	http://xmlns.com/foaf/0.1/name
-101	http://xmlns.com/foaf/0.1/age
-102	http://xmlns.com/foaf/0.1/homepage

RDF

CoRAL can then be used as a (very basic) substitute for RDF.

For example, the RDF graph (in Turtle syntax)

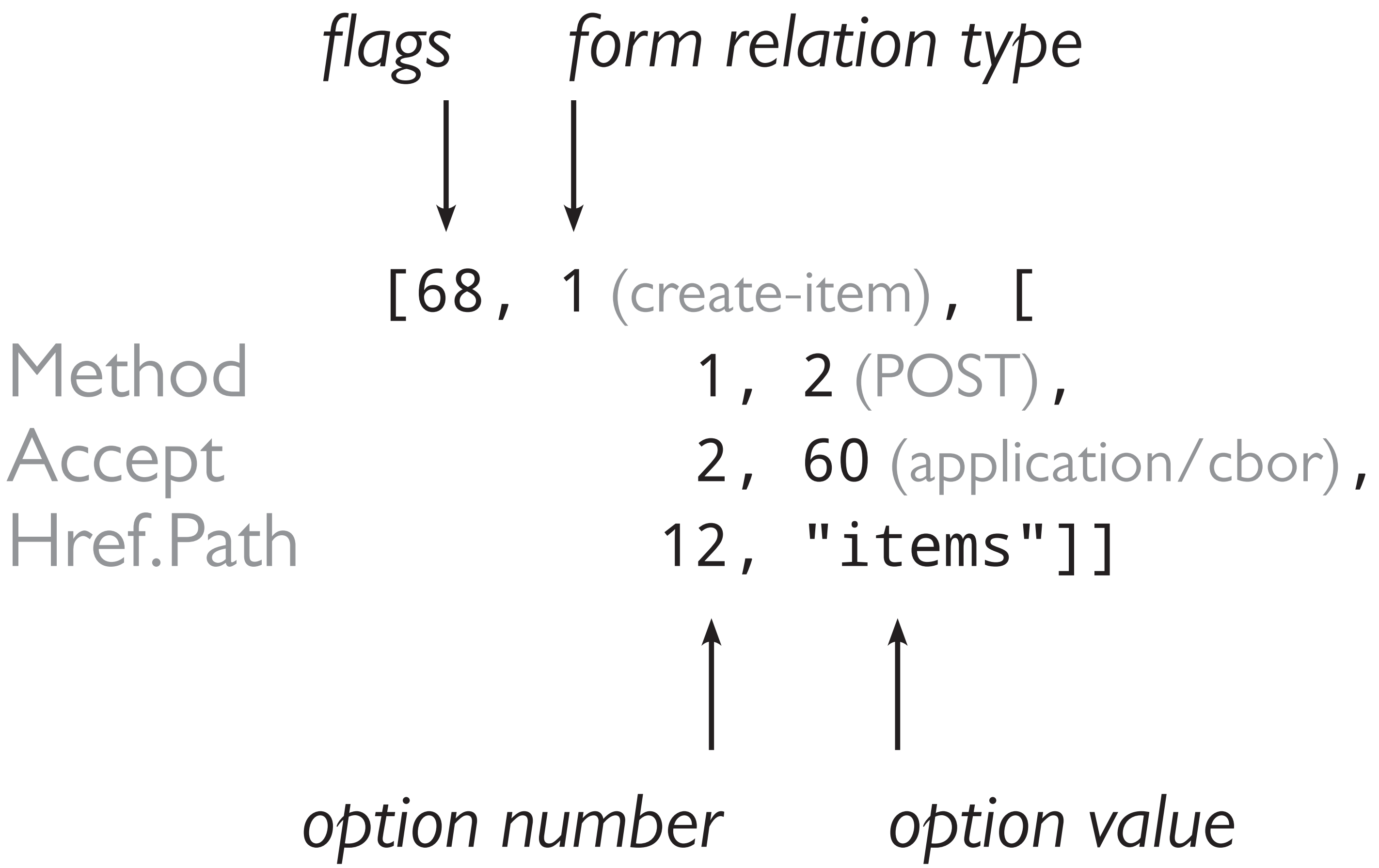
```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<> foaf:name     "John Doe" ;
   foaf:age      32 ;
   foaf:homepage <coap://doe.example/> .
```

could be serialized in CoRAL as follows:

```
[12, -100 (name),      [3, 0 (text/plain)   ], «John Doe»]
[12, -101 (age),       [3, 9 (uint8)        ], «32»]
[ 0, -102 (homepage),  [4, "coap", 6, "doe.example"]]
```

Forms



flags *form relation type*

```
                [68, 1 (create-item), [
Method                   1, 2 (POST),
Accept                   2, 60 (application/cbor),
Href.Path               12, "items"]]
```

option number *option value*

Editing

The representation of a resource typically contains a form that allows to edit the resource. However, it may be more efficient to include this form in a representation that links to the resource.

CoRAL defines two flags for this:

- Setting the *Updateable* Flag in a link defines a default form (`method="PUT"`) that can be used to update the target resource.
- Setting the *Deleteable* Flag in a link defines a default form (`method="DELETE"`) that can be used to delete the target resource.

https://tools.ietf.org/html/draft-hartke-t2trg-coral

Thank you!

Photos by Daniel Smith
https://flic.kr/p/swmhQg
https://flic.kr/p/swokBi